Remarks

Claims 1-21 are pending in the current application. Claims 1-18 stand rejected under 35 U.S.C. §103(a) as being unpatentable over *Dupay, et al.* (U.S. Patent No. 6,253,215) in view of *Agesen, et al.* (U.S. Patent No. 6,253,215). Claims 1, 13 and 14 have been amended to correct typographical errors.

Elections/Restrictions

In the Action, the Examiner requires restriction under 35 U.S.C. §121 to one of the following:

> Group I -- Claims 1-18 drawn to source to source code translation, classified in class 717, subclass 137.
>
> Group II – Claims 19-21 drawn to simulation modeling, classified in class 703, subclass 22.

A provisional election was made with traverse by Applicants on March 3, 2004 by telephone to prosecute Group I, Claims 1-18. Applicants herein affirm the election of Group I Claims 1-18. However, Applicants respectfully request reconsideration of the application as filed and traverse the Examiner's restriction requirement as follows:

The Examiner argues that the inventions are distinct because the inventions, as shown above, are related as combination and subcombination. The MPEP requires two criteria to be met by an Examiner for a proper restriction between patentably distinct inventions 1) the inventions must be independent or distinct as Claimed; and, 2) there must be a serious burden on the examiner if restriction is required. *See* MPEP § 803.

The Examiner states the criteria under 806.05(c) to establish that the asserted combination and subcombination inventions are distinct. To support a requirement for restriction, the MPEP sets forth that two-way distinctness and the reasons for insisting on restriction are necessary (i.e., separate classification, status, or field of search). *See* MPEP § 806.05(c). Applicants respectfully submit that although the Examiner has explained that the invention falls within separate classifications, it would not be an undue burden placed upon the Examiner to examine all the Claims as filed. Applicants respectfully suggest that the method Claimed in Claims 1-18 have a common core structure and can be easily searched and examined in conjunction with Claims 19-21 which relate to a computer system for simulation modeling generally utilizing the method of Claims 1-18. Therefore, Applicants respectfully request reconsideration of the restriction requirement.

Based upon Applicants provisional election of Group I Claims 1-18 and subsequent examination thereof, Applicants respond to the Examiner's rejections as follows:

Claim Rejections – 35 U.S.C. §112

Claim 14 stands rejected under 35 U.S.C. §112 as having insufficient antecedent basis for the limitation of "*...wherein the step of developing an object data structure...*" found within the Claim. In response, Applicants have amended Claim 14 to remove a typographical error. Applicants respectfully submit that Claim 14, as now amended, is allowable.

Claim Rejections – 35 U.S.C. §103

Claims 1-18 stand rejected under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent No. 6,523,171 to Dupuy *et al* ("*Dupuy*") in view of U.S. Patent No. 6,253,215 to Agesen *et al* ("*Agesen*"). The Examiner cites *Dupuy* as disclosing the features of Applicants' independent Claim 1 with the exception of the element of "developing object oriented

extensions, wherein an existing application of the non-object oriented computer environment remains executable and wherein the new object oriented computer environment accesses information of the non-object oriented computer environment." *Office Action*, p. 4. However, the Examiner states that *Agesen* discloses the element of Applicants' Claim 1 that is absent from *Dupuy* and that it would have been obvious to one having ordinary skill in the art to incorporate the teaching of *Agesen* into the teaching of *Dupuy*. The Examiner reasons that it would be necessary to do so "because one would want to provide a flexible system which can access both types of environments." *See, Office Action*, pp. 4-5. Applicants respectfully submit, however, that neither *Dupuy* nor *Agesen*, singularly nor in combination, teach, suggest, or render obvious the features of independent Claim 1.

The Federal Circuit has held that all Claim limitations must be considered in order to make a proper 103 combination of prior art. A reference does not render the Claimed combination *prima facie* obvious if a material limitation has been ignored. *In re Fine*, 837 F.2d 1071, 5 U.S.P.Q.2d 1596 (Fed. Cir. 1988); *Also See, In re Evanega*, 829 F.2d 1110, 4 U.S.P.Q.2d 1249 (Fed. Cir. 1987). The Federal Circuit has also held that obviousness cannot be established by combining the teachings of the prior art to produce the Claimed invention absent some teaching, suggestion, or incentives purporting the combination. *ACS Hospital Systems, Inc. v. Montefiore Hospital*, 732 F.2d 1572, 221 U.S.P.Q. 929, 933 (Fed. Cir. 1984); *Also See, In re Geiger*, 815 F.2d 686, 2 U.S.P.Q.2d 1276, 1278 (Fed. Cir. 1987). Absent some showing in the prior art, the Examiner has impermissibly used the Applicant's teaching to search the prior art for the Claimed elements and combine them as taught by the inventor. *In re Zurko*, 111 F.3d 887, 42 U.S.P.Q.2d 1476 (Fed. Cir. 1997). "To support the conclusion that the Claimed invention is directed to obvious subject matter, either the references must expressly or impliedly suggest the

Claimed invention or the Examiner must present a convincing line of reasoning as to why the artisan would have found the Claimed invention to have been obvious in light of the teachings of the references." *Ex parte Clapp*, 227 U.S.P.Q. 972, 973 (Bd. Pat. App. & Inter. 1985).

A proper §103 combination cannot be made if the art of record "teaches away" from the combination. In short, teaching away is the antithesis of the art suggesting that a person of ordinary skill go in the Claimed direction. Teaching away from the art is a *per se* demonstration of lack of *prima facie* obviousness. *In re Dow Chemical Co.*, 837 F.2d 469, 5 U.S.P.Q.2d 1529 (Fed. Cir. 1988); *In re Fine*, 837 F.2d 1071, 5 U.S.P.Q.2d 1596 (Fed. Cir. 1988); *In re Nielson*, 816 F.2d 1567, 2 U.S.P.Q.2d 1525 (Fed. Cir. 1987). A §103 rejection based on a modification of a reference that destroys the intent, purpose or function of the invention disclosed in the reference, is not proper and the *prima facie* case of obviousness cannot be properly made. *In re Gordon*, 733 F.2d 900, 221 U.S.P.Q. 1125 (Fed. Cir. 1984).

With respect to Claim 1, there is no basis in the art for combining the references as suggested by the Examiner.

As to Claim 1, the Examiner states that *Dupuy* in col. 3, lines 19-22 teaches "defining requirements for the new object oriented computer environment". *Office Action,* p. 4. Applicants respectfully disagree. Claim 1 requires the element of "defining requirements for the new object oriented computer environment." The specification gives an example of defining requirements by creating a requirements document. In this example, the requirements document includes such elements as the enhancements from the conversion of a non-object oriented environment to an object oriented environment, and improved functionality or future functionality of the non-object oriented environment.

Under §103, *Dupuy* must teach or suggest this element of Applicants' Claim. However, *Dupuy* does not teach or suggest "defining requirements." In contrast, the invention of *Dupuy* is merely a translator that takes a program written in a non-object oriented language and translates it into a program capable of being executed in an object oriented language. *Dupuy* does not set out to define requirements or create a requirements document for an object oriented environment. At most, the parser in *Dupuy* understands the requirements already required by an object oriented language. Nothing in col. 3, lines 19-22 nor any other portion of *Dupuy* discloses the Claim element. In fact, col. 3, lines 19-22 only disclose that the parser of *Dupuy* is capable of recognizing the grammar and syntax of the non-object oriented language as opposed to creating or defining the requirements of the new object oriented environment. Therefore, Applicants submit that *Dupuy* does not teach or suggest this feature of Applicants' invention

The Examiner also states that *Dupuy* in col. 3, lines 27(sic)-26, Fig. 2, and col. 4, lines 15-52 teaches "preparing the new object oriented environment, wherein the new object oriented computer environment includes the requirements, the grammar and syntax and object oriented extensions.". *Office Action*, p.4. Applicants respectfully disagree. Claim 1 requires the element of "preparing the new object oriented environment, wherein the new object oriented computer environment includes the requirements, the grammar and syntax and object oriented extensions." The specification gives an example of this Claim element. In the example given, actual code is generated to create the required specifications, grammar, syntax and object oriented extensions that are part of the new object oriented environment. Grammar, syntax and extensions are requirements that extend beyond and control many features of the new object oriented environment. These are far beyond the mere generation of classes, for instance. §103 requires that *Dupuy* must teach or suggest this element of Applicants' Claim, but *Dupuy* fails to do so.

The invention of *Dupuy* is merely a translator which translates a program written in a non-object oriented language into a program executable in an object oriented language. In contrast to Applicants' Claim element, the parser of *Dupuy* only creates new classes of objects for the non-object oriented source code, it does not generate grammar, syntax and extensions as required by the Claim element. At best, the parser of *Dupuy* recognizes the requirements, and the syntax and grammar rules required to create the object. In the lines cited by the Examiner, *Dupuy* only discloses building, from portions of the non-object oriented source code, an object model with the object oriented code as opposed to generating the required specifications, grammar, syntax and extensions of an object oriented environment. Nothing in *Dupuy* discloses or suggests this feature of Applicants' invention. Therefore, Applicants submit that *Dupuy* does not teach or suggest this feature of Applicants' invention.

The Examiner states that *Agesen* at col. 2, lines 15-26 teaches "developing object oriented extensions, wherein an existing application of the non-object oriented remains executable and wherein the new object oriented computer environment accesses information of the non-object oriented computer environment." Applicants respectfully disagree. Claim 1 requires the element of "developing object oriented extensions, wherein an existing application of the non-object oriented computer environment remains executable and wherein the new object oriented computer environment accesses information of the non-object oriented computer environment." In general, an object oriented extension is known in the art as a mechanism that enables a runtime environment to find and load extension classes without the extensions having to be named in the class path. The specification provides an example of developing object oriented extensions. In one example, a header structure and a data structure is created to reference a non-object oriented data structure. In that example, the header allows the new object

oriented computer environment to access a non-object oriented data structure in the same way as any other object oriented extension. In contrast, *Agesen* discloses a memory management device which clears unused memory for a program which contains both native code and target code and has separate memory stacks. Nothing in col. 2, lines 15-26 or any other portion of *Agesen* discloses the above mentioned element of Applicants' invention. In col. 2, lines 15-26 cited by the Examiner, *Agesen* discloses only that object oriented programs may use features of an operating system that are not available in a virtual machine. But the operating system of *Agesen* is not the same thing as the non-object oriented computer environment of the Claim. Usage of features unique to an operating system by a virtual machine has no relation to developing object oriented extensions such as header and data structures to reference a non-object oriented data structure in an object oriented environment as required by the Claim. Therefore, applicants submit that *Agesen* does not teach or suggest this feature of Applicants' invention.

Applicants also respectfully submit that there is no suggestion to combine the references of *Dupuy* and *Agesen*. The burden is on the Examiner to show some teaching, suggestion or incentives purporting the combination of the two references. Applicants respectfully submit that the Examiner has not met this burden. *Dupuy* discloses a translator which translates non-object oriented source code to object oriented source code and *Agesen* discloses a memory management device which uses a garbage collector to clear memory from native and target stacks. There is no motivation on the face of the two references to combine a source code translator with a device that manages garbage collection for stack memories. Since the references do not have any relationship to on each other there is no suggestion for the combination much less an incentive.

Dupuy, on its face, teaches away from Applicant's invention. Applicant's invention allows non-object oriented programs to be used and reused in runtime. The non-object oriented

source code in Applicant's invention is "encapsulated". For example, Claim 1 requires "developing object oriented extensions, wherein an existing application of the non-object oriented computer environment *remains executable* and when the new object oriented computer environment *accesses information* of the non-object oriented computer environment." Put simply, the non-object oriented computer environment of a computer program remains viable in runtime. On the other hand, *Dupuy* is a translator. The non-object oriented code is translated and then discarded. It is not active at compile time. It is not active at runtime. A translator teaches away from Applicant's invention of encapsulations through creation of extensions for the non-object oriented computer environment or program.

Furthermore, *Agesen,* on its face, teaches away from Applicants' invention. The specification of the instant application describes a method of incorporating *any* non-object oriented environment into a new object oriented environment. In one example, this new object oriented environment would include all of the programs, features and functions of the non-object oriented environment. In contrast, *Agesen* drastically limits its disclosure to only a few possible candidate programs. For instance *Agesen* states that it "may not be suitable for all functions of a system or it may not be economically feasible to covert all of the programs in an existing legacy system into object-oriented programs." Since *Agesen* suggests and in fact recites that not all programs should be converted, *Agesen*'s teachings directly conflict with Applicants' invention of converting the entire non-object oriented environment to an object oriented environment wherein all programs are converted and indeed reused. Therefore, *Agesen* teaches away from Applicant's invention.

The intent, purpose and function of each of the *Dupuy* and *Agesen* references is destroyed by the proposed § 103 combination of the Examiner and therefore, cannot survive as a *prima*

facie case of obviousness. For example, *Dupuy* discloses a translator. On the other hand, *Agesen* discloses a memory management device which remains active to clear unused memory or programs that contain both native code and target code. The two cannot coexist. The translator of *Dupuy* reads, changes, and rewrites native code or non-object oriented code into an object oriented format. The native code or object oriented code is then *discarded.* There is no reason for function of *Agesen* to monitor memory usage of code that has been discarded. In fact, the utility of the *Agesen* invention is destroyed if there is no native code for which to manage or clear memory. On the other hand, the translation functions of *Dupuy* are unnecessary if the native code is running as is required by *Agesen.* Therefore, the functionality of both the *Agesen* and *Dupuy* inventions are destroyed by the combination suggested by the Examiner. For this reason, the references cannot be combined in a proper § 103 rejection.

Since *Dupuy* does not teach all of the elements of as suggested by the Examiner and since *Agesen* does not teach the missing elements of Claim 1, the two combined do not teach all of the elements of Claim 1. Also, there is no suggestion or motivation in *Dupuy* or *Agesen* to combine the elements as suggested by the Examiner. Furthermore, both *Dupuy* and Agesen teach away from the Claimed invention and so are not properly combinable as § 103 references. Moreover, *Dupuy* cannot be combined with *Agesen* nor can *Agesen* be combined with *Dupuy* because their functionality would be destroyed through the combination. For all of these reasons, Applicants respectfully suggest that Claim 1 is allowable over the art of record.

As to Claim 2, the Examiner states that *Dupuy* discloses identifying a commercially available object oriented computer environment (col. 3 lines 6-10). Applicants respectfully disagree. Claim 2 requires the element of identifying an object oriented computer environment including identifying a commercially available object oriented computer environment. The

specification provides examples of identifying the commercially available object-oriented computer environment. In that example, a commercially available object-oriented computer environment is selected and the benefits of the commercially available object-oriented computer environment are identified. Furthermore, the example in the specification may base the selection of the commercially available object-oriented computer environment on the compatibility of the non-object oriented computer environment. In contrast, *Dupuy* only selects the Java language as a target source code with no consideration to the benefits or compatibility. Nothing in col. 3 lines 6-10 or any other portion of *Dupuy* discloses the above mentioned element of Applicants' invention. In lines 6-10, *Dupuy* discloses only an example of translating an application written in an non-object oriented environment into an application written in the Java language. Therefore, applicants submit that *Dupuy* does not teach or suggest this feature of Applicants' invention.

The above argument not withstanding, Claim 2 also depends upon Claim 1 and also includes all of the elements of Claim 1 as well as the Claim element of identifying a commercially available object oriented computer environment. Since the prior art does not disclose all of the elements of Claim 1, the prior art cannot therefore disclose all of the elements of dependent Claim 2.

As for Claim 3, the Examiner states that *Dupuy* discloses identifying a legacy non-object oriented environment (col. 3 lines 6-10). Applicants respectfully disagree. Claim 3 requires the Claim element of identifying a non-object oriented computer environment including identifying a legacy non-object oriented computer environment. The specification provides examples of identifying a legacy non-object oriented computer environment. In this example, a legacy non-object oriented language is selected and its features are identified. These features are identified

for the purpose of retaining the benefits of the features after the non-object oriented computer environment is converted to an object-oriented computer environment. In contrast, *Dupuy* only selects a non-object oriented language on which an application is written for in preparation for translation to an application for an object oriented language. Nothing in the lines cited by the Examiner or throughout *Dupuy* selects a legacy non-object oriented language and identifies its features that wish to be retained. Therefore, Applicants submit that *Dupuy* does not teach or suggest this feature of Applicants' invention.

The above argument not withstanding, Claim 3 depends upon Claim 1 and includes all of the elements of Claim 1. Since the prior art does not disclose all of the elements of Claim 1, the prior art cannot disclose all of the elements of dependent Claim 3

As for Claims 4-8, the Examiner states that *Dupuy* does not explicitly disclose the legacy non-object computer environment, including a user language interface and data structure, allowing multiple users, including a distributed environment, allowing simulation modeling for the analysis of the performance software. However, the Examiner takes official notice that a user language and data structure, allowing multiple users, including a distributed environment, and allowing simulation modeling for analysis of the performance software were well known in the art at the time the invention was made.

The Examiner may take official notice of facts outside of the record which are capable of instant and unquestionable demonstration as being "well-known" in the art *In re Ahlert, 424 F.2d 1088, 1091.* When a rejection is based on facts within the personal knowledge of the examiner, the data should be stated as specifically as possible, and the facts must be supported, when called for by the applicant, by an affidavit from the Examiner. MPEP § 2144.03.

Applicants respectfully submit that an official notice is improper because a user language and data structure, allowing multiple users, including a distributed environment, and allowing simulation modeling for analysis *in the context of the Claims* are not "well known". Applicant also respectfully submits that the facts are not supported or stated with sufficient specificity to support a rejection of the Claims.

Applicant requests an affidavit from the Examiner supporting the alleged facts assumed to be well known which form the basis for any rejection.

The above notwithstanding, Claims 4-8 depend upon Claim 1 and includes all of the same elements of Claim 1. Since the prior art does not disclose all of the elements of Claim 1, the prior art in conjunction with official notice of other elements of the dependent claims cannot disclose all of the elements of Claims 4-8.

As to Claim 9, the Examiner states that *Dupuy* discloses selecting semantics of the non-object oriented environment (col. 2 lines 19-22, 27-35). However, Claim 9 depends upon Claim 1 and includes all of the elements of Claim 1. Since the prior art does not disclose all of the elements of Claim 1, the prior art cannot disclose all of the elements of dependent Claim 9.

As to Claim 10, the Examiner states that *Dupuy* discloses "selecting semantics compatible to the non-object oriented computer environment" (col. 3 lines 19-22, 27-35). Applicants respectfully disagree. Claim 10 requires the element of "selecting semantics compatible to the non-object oriented environment." The specification states that compatibility of the selected semantics is necessary as the non-object oriented language is to be extended to provide the new object oriented capabilities. In contrast, the parser of *Dupuy* only recognizes the semantics of the non-object oriented language and does not "select semantics compatible to the non-object oriented environment" such that the non-object oriented language is extended to

provide the new object oriented capabilities. Therefore, Applicants respectfully submit that *Dupuy* does not teach or suggest this feature of the invention.

As to Claim 11, the Examiner states that *Dupuy* discloses selecting the semantics of the existing object oriented computer environment (col. 2 lines 27-35, Fig. 2 and col. 4 lines 15-52). However, Claim 11 depends upon Claim 1 and includes all of the elements of Claim 1 plus an additional element. Since the prior art does not disclose all of the elements of Claim 1, the prior art cannot disclose all of the elements of dependent Claim 11.

As to Claims 12, the Examiner states that although *Dupuy* does not explicitly disclose developing object oriented extensions includes developing an object header structure and object data structure, the Examiner takes Official Notice that an object header structure, and an object data structure were well known in the art at the time the invention was made. However, "assertions of technical facts in areas of esoteric technology must always be supported by citation of some reference work." *In re Barr,* 444 F.2d 588. Applicants respectfully submit that developing object oriented extensions includes developing an object header structure and object data structure as required by the Claim cannot be cited as Official notice absent a further showing in a reference. Also, Claim 12 depends upon Claim 1 and includes all of the elements of Claim 1 plus an additional element. Since the prior art does not disclose all of the elements of Claim 1, the prior art cannot disclose all of the elements of dependent Claim 12.

As to Claim 13, the Examiner failed to make a proper rejection. Claims 12 and 13 have been rejected simultaneously based on the same logic. However, this rejection fails to present any reason how the element of "wherein the step of developing an object header structure includes developing an object header structure that provides a unified object oriented interface to a user and internal objects" required by Claim 13 is found in the prior art. Since this element is

not in the prior art and has not been asserted to be in the prior art, it appears that Claim 13 is allowable.

As to Claim 14, the Examiner states that *Dupuy* discloses developing an object data structure containing a data structure of the non-object oriented computer environment (col. 3, lines 37 – col. 4 line 7). Applicants respectfully disagree. Claim 14 requires the element of "developing an object data structure containing a data structure of the non-object oriented environment." The specification gives examples of such data structures. The example includes creating various objects for the functions which were part of the non-object oriented environment. These objects include functions such as client workload, etc. In contrast, the parser of *Dupuy* only creates various object classes for the non-object oriented program and does not create data structures of the non-object oriented environment such as Applicants' invention as required by the Claim. Therefore, Applicants respectfully submit that *Dupuy* does not teach or suggest this feature of the invention.

As to Claim 15, the Examiner states that *Dupuy* discloses developing general purpose utility classes (col. 4 lines 22-25). However, Claim 15 depends upon Claim 1 and includes all of the elements of Claim 1. Since the prior art does not disclose all of the elements of Claim 1, the prior art cannot disclose all of the elements of dependent Claim 15.

As to Claim 16, the Examiner states that *Dupuy* discloses creating new code (col. 3 lines 27-26, Fig. 2 col. 4 lines 15-52, col. 4 lines 22-25). However, Claim 16 depends upon Claim 1 and includes all of the elements of Claim 1. Since the prior art does not disclose all of the elements of Claim 1, the prior art cannot disclose all of the elements of dependent Claim 16.

As to Claim 17, *Dupuy* does not disclose "wherein the step of preparing the new object oriented computer environment includes creating an operating system". The Examiner states

that *Dupuy* in col. 3, lines 27(sic)-26, Fig. 2 and col. 4, lines 15-52 discloses Claim 17's required element of "creating an operating system". Nothing in *Dupuy* discloses creating an operating system. Instead, col. 3, lines 27(sic)-26, Fig. 2 and col. 4, lines 15-52 discloses creating various object classes from a non-object oriented programs. Therefore, Applicants respectfully submit that *Dupuy* does not teach or suggest the element of Claim 17.

As to Claim 18, the Examiner states that *Dupuy* discloses wherein the new object oriented computer environment includes an object oriented computer language (col. 3 lines 4-15). *Dupuy* discloses an object oriented computer language; however, Claim 18 also depends upon Claim 1 and includes all of the elements of Claim 1. Since the prior art does not disclose all of the elements of Claim 1, the prior art cannot disclose all of the elements of dependent Claim 18.

Conclusion

In view of the foregoing differences between Applicants' invention and the cited references, singularly or in combination, Applicants submit that the presently Claimed invention is patentably distinguishable. Applicants have made an earnest attempt to place this Application in condition for allowance.

Applicants respectfully request a reconsideration of the application and earnestly solicit allowance. Should it facilitate allowance of the application, the Examiner is invited to telephone the undersigned attorney.

Respectfully submitted,

Dated: SEPT 16 2004

By: _____

George R. Schultz
Reg. No. 35,674
SCHULTZ & ASSOCIATES, P.C.
ONE Lincoln Centre
5400 LBJ Freeway, Suite 525
Dallas, Texas 75240
(214) 210-5940 telephone
(214) 210-5941 facsimile